Exercício Programa II : Integração numérica de Gauss

MAP3121 - Métodos Numéricos e Aplicações

Prof. Dr. Pedro Peixoto

5 de junho, 2022

Laura do Prado Gonçalves Pinto
nºUSP 11819960

Mateus Stano Junqueira
nºUSP 11804845

Lista de Figuras

Conteúdo

1 Introdução

Neste relatório é apresentado todo o processo de construção lógica e computacional da resolução do segundo exercício programa proposto para a disciplina MAP3121- Métodos Numéricos e Aplicações assim como os resultados obtidos pelos métodos apresentados e as conclusões tomadas sobre a tarefa proposta.

No exercício proposto trabalha-se a implementação de um algorítimo para resolução de cálculos de integrais duplas utilizando a ferramenta conhecida como fórmulas de integração de Gauss. Para tal tarefa, lança-se mão dos de códigos efetuados para a resolução de integrais segundo o método de Gauss em um intervalo específico de [-1,1] com ferramentas adicionais para a mudança de variáveis quando necessário para que este seja válido a quaisquer intervalos desejados. Em seguida, prova-se a confiabilidade do código com a resolução de 4 modos, no caso 4 exercícios propostos ao final da tarefa, os quais dedicam-se a efetuar cálculos em variadas situações.

Este relatório consta a metodologia utilizada para se alcançar tais objetivos além de todos os recursos utilizados para a construção do algoritmo assim como as referentes análises sobre os resultados encontrados.

2 Descrição

2.1 Análise do problema

O problema proposto na tarefa deste primeiro exercício programa consiste em utilizar um método de análise numérica para resolução de integrações por meio das chamadas fórmulas de integração de Gauss. Tais fórmulas consistem em regras de quadratura que aproximam uma integral definida de uma função soma ponderada de valores específicos dentro do domínio da integração. O objetivo central deste exercício programa é a implementação de tal ferramenta para resolução de integrais duplas em regiões R do plano por fórmulas iteradas com o método de Gauss com *n* nós e pesos fornecidos para o intervalo [-1, 1].

2.2 Quadratura Gaussiana

Uma regra de quadratura Gaussiana consiste em uma fórmula construída para se obter o resultado exato para polinômios de grau *2n-1* ou menores por meio de uma escolha adequada de nós x_i e pesos w_i para i de 1 à n.

Como mencionado anteriormente, a fórmula de Gauss se traduz em uma soma ponderada cujos pesos possibilitam que por meio dos nós não necessariamente igualmente espalhados chegue-se a uma solução exata para a integral em questão por meio da expressão:

$$\int_{-1}^{1} f(x)dx \approx \sum_{i-1}^{n} w_i f\left(x_i\right) \tag{1}$$

O método da quadratura Gaussiana trabalha como escolher tais nós e tais pesos de forma a conseguir a melhor aproximação possível da integral desejada. Como nessa ferramenta trabalha-se com pontos e pesos não iguais em espaçamento como em outros métodos temons *n* pontos x_i e de tal forma *n* pesos w_i a serem determinados para resolução completa da integral referida. Assim, trabalhamos com 2n graus de liberdade na resolução desse sistema.

Quando desejamos que uma fórmula seja aproximada a um determinado valor, deseja-se que ela seja exata à uma família de funções e, para este caso usaremos os polinômios. Assim, para solução do cálculo desejado aproximaremos a função por um polinômio de grau 2n-1.

O grau 2n-1 pode ser explicado pelo simples fato de que para tal equação o número de incógnitas é também igual 2n, ou seja o mesmo grau de liberdade que a função a qual desejamos aproximá-lo de forma exata.

Para o intervalo de intervalo de integração desejado, a obtenção dos pontos se dá por meio das raízes de um polinômio de Legendre $p_n(x_n)$ por meio dos quais pode-se obter os pesos referentes a cada um deles por meio da equação:

$$c_i = \int_{-1}^{1} \prod_{j=1, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} dx \tag{2}$$

Se temos um espaço vetorial C[a,b] onde estão contidas todas as equações dos polinômios mencionados, no qual é possível definir um produto vetorial tal qual

$$\langle f, g \rangle = \int_{a}^{b} f(x)g(x)dx \tag{3}$$

Incia-se então com as funções polinomiais *1*,x, ...,x^n , até o n-ésimo polinômio e aplica-se o método de ortogonização de Gram-Schmidt nessa família criando uma nova leva de n polinômios. Estes polinômios são os polinômios ortogonalizados de Legendre.

2.3 Mudança de variável na integração

A maioria das resoluções e dos cálculos para os nós e pesos no caso discutido até agora pode ser encontrado em tabelas, porém uma questão específica para utilização dessa resolução específica é que o intervalo de integração deve ser [-1,1].

Desta forma, para ser possível realizar esse método será necessário efetuar uma mudança de variável para manipular a integral de valor equivalente de um intervalo qualquer [a,b] para o intervalo desejado [-1,1]. Para isso, precisamos pensar em uma função simples que efetue essa equivalência, no caso uma reta que passe por ambos pares de pontos. Assim chega-se em:

$$y = -1 + \frac{2}{b-a}(x-a) \tag{4}$$

trocando x por uma variável u igual a

$$u = -1 + \frac{2}{b-a}(x-a) \tag{5}$$

$$du = \frac{2}{b-a} \cdot dx \tag{6}$$

fazendo as substituições e manipulações necessárias chega-se na seguinte equação para os novos intervalos:

$$x = \frac{b+a}{2} + \frac{b-a}{2} \cdot u \tag{7}$$

Assim a nossa transformação de intervalos será tal que os novos limites e a função no intervalo desejado a ser integrado fica:

$$\int_2^b f(x)dx = \int_{-1}^1 f\left(\frac{b+a}{2} + \frac{b-2}{2} \cdot u\right) \cdot \frac{b-a}{2} du \tag{8}$$

Assim, efetuada a troca demonstrada é possível efetuar a integração pelo método de quadratura gaussiana.

Essa mudança resulta ainda em polinômios exatos até grau cinco visto que para resolução no intervalo [-1,1] pelas fórmulas gaussianas é exata para todo polinômio de grau menor ou igual 2n-1 e, a equação deduzida para essa operação foi feita para o grau n=2 (para dois pontos a e b).

2.4 Integração dupla pelas fórmulas de Gauss

Quando desejamos efetuar uma integral dupla em uma região R do plano , a qual em x está delimitada por duas variáveis [a,b] e em y por duas funções [c(x),d(x)] ou vice-versa. Desta forma podemos usar o cálculo de integrações iteradas por meio das fórmulas de Gauss para n nós tal que:

$$I = \sum_{i=1}^{n} u_i F(x_i) \tag{9}$$

e

$$F\left(x_i\right)=\sum_{j=1}^{n} v_{ij} f\left(x_i, y_{ij}\right) \tag{10}$$

3 Implementação do Código

3.1 Algoritmo

O algoritmo desejado, descrito no enunciado do exercício programa, pede que se implemente um método para o cálculo de integrais duplas por meio das fórmulas de integração de Gauss no intervalo de [-1,1], incluindo métodos para a mudança de variável nos limites de integração para que seja possível a utilização do mesmo método em qualquer intervalo desejado e seja possível efetuar os cálculos com o mesmo método do intervalo fixo mencionado. Em seguida deve-se testar a eficácia e acurácia do algorítimo em 4 exemplos diferentes sobre a resolução de integrais em regiões R do espaço.

De forma geral, é demandado no exercício que sejam criadas funções para a resolução de integrações utilizando o método de quadratura de Gauss, uma para a mudança do intervalo de variável para que o intervalo de integração sempre esteja no determinado [-1,1] e a ampliação da quadratura para sua utilização em integrais duplas no espaço R onde uma variável no espaço está determinado por duas funções e outra por duas variáveis.

Tarefa

Implemente um programa para o cálculo de integrais duplas em regiões R do plano, como as descritas acima, por fórmulas iteradas. Você deverá usar fórmulas de Gauss com n nós para as integrações numéricas. Os nós e os pesos são fornecidos para o intervalo $[-1, 1]$ e o programa deverá fazer os ajustes necessários para outros intervalos. *Use precisão dupla.* No numpy é o padrão e em C use DOUBLE.

Teste o seu programa nos exemplos abaixo usando fórmulas de Gauss com $n = 6$, 8 e 10. Os nós e os pesos estão no arquivo *dados.txt* (devido à simetria já mencionada, são dados apenas os nós maiores ou iguais a zero).

Figura 1: Enunciado Exercício Programa II

3.2 Metodologia aplicada

O algorítimo aplicado para a resolução desse exercício programa segue a seguinte linha de raciocínio:

- Em primeiro cria-se uma função capaz de efetuar a quadratura de Gauss para resoluções de integrais para o intervalo [-1,1]
- Implementa-se um código para efetuação de troca de variável na integração para que qualquer intervalo de integração possa ser resolvido como uma integral definida em [-1,1].
- código para resolução dos exemplos mencionados após o enunciado da tarefa

A seguir será destrinchado cada um destes seguimentos de raciocínio, numerados em modos 0,I, II, III, e IV detalhando as funções, ferramentas e lógicas envolvidas em seu desenvolvimento assim como os resultados obtidos em cada um.

3.3 Modo 0

O assim chamado "Modo 0"neste exercício é o código dedicado a efetuar a quadratura de Gauss para uma integral dupla numa região R do plano , em um intervalo de integração qualquer.

Para executar esse cálculo, criou-se uma função **def integral dupla** apresentada abaixo:

```python
def integral_dupla(f, a, b, pontos_e_pesos_x, c, d, pontos_e_pesos_y):
    """
    f: função de x e y a ser integrada
    a: limite inferior do intervalo de integração em x
    b: limite superior do intervalo de integração em x
    pontos_e_pesos_x: pontos e pesos para o método de Gauss em x (deve ser da forma [(ponto1, peso1), (ponto2, peso2), ...].)
    c: limite inferior do intervalo de integração em y (função de x)
    d: limite superior do intervalo de integração em y (função de x)
    pontos_e_pesos_y: pontos e pesos para o método de Gauss em y (deve ser da forma [(ponto1, peso1), (ponto2, peso2), ...].)

    """

    I = 0
    for x_i, w_i in pontos_e_pesos_x:
        # troca de variavel de x para o intervalo [a,b]
        x_i = (a + b) / 2 + (b - a) * x_i / 2
        F = 0
        for y_ij, w_ij in pontos_e_pesos_y:
            # troca de variavel de y para o intervalo [c(x_i),d(x_i)]
            y_ij = (c(x_i) + d(x_i)) / 2 + (d(x_i) - c(x_i)) * y_ij / 2
            F += w_ij * f(x_i, y_ij) * (d(x_i) - c(x_i)) / 2
        I += w_i * F * (b - a) / 2

    return I
```

Esta função é basicamente a aplicação das equações elaboradas anteriormente para o código. Inicialmente efetua-se a mudança de variável para que o intervalo de integração esteja sempre em [-1,1]. Para tal utiliza-se para ambos intervalos a equação 7 para x e y de forma que em seguida seja possível utilizar as fórmulas 9 e 10 para realizar a quadratura Gaussiana.

O funcionamento do restante do modo é relativamente simples e segue os comentários. Ele consiste em demandar do usuário as entradas requeridas para o funcionamento da função integral dupla, ou seja , recebe e trata as entradas para a função a ser integrada, os limites inferior e superior [a,b] em x da função, os limites inferior e superior em função de x [c(x),d(x)] de y, e os respectivos nós e pesos em x e y.

Vale ressaltar que o número de nós deve ser indicado também nessa parte do programa, sendo este tanto para x quanto para y uma escolha entre 6, 8 ou 10 nós como exigido no enunciado deste exercício programa.

3.4 Modo I

Neste modo trabalha-se a resolução do exemplo I dado após o enunciado da tarefa deste programa, como segue:

Exemplo 1 Calcule os volumes do cubo cujas arestas tem comprimento 1 e do tetraedro com vértices $(0,0,0)$, $(1,0,0)$, $(0,1,0)$ e $(0,0,1)$. Você deve obter resultados exatos, exceto por erros de arredondamento (por que?).

Figura 2: Enunciado Exemplo I

Na primeira parte, como estamos calculando o volume de uma forma geométrica específica, ao selecionar modo I já coloca-se que a função a ser integrada será $f(x,y) = 1$, os intervalo [a,b] em x e [c,d] em y são ambos [0,1]. Então, há três opções a se seguir a depender da quantidade de nós escolhidas para a resolução (6, 8 e 10), como segue em anexo:

```
elif modo == 1:
        """Modo 1 - Resolve o Exemplo 1"""

        # Cubo --------------------------------------------------------------------------

        print("\n\nCálculo do volume do cubo cujas arestas tem
    comprimento 1:\n")
        print("Função a ser integrada: f(x,y) = 1\n")
        print("Intervalos de integração: \n a = 0, b = 1, c = 0, d =
     1\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
                pontos_e_pesos_x=n6,
                c=lambda x: 0,
                d=lambda x: 1,
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 -->  I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
                pontos_e_pesos_x=n8,
                c=lambda x: 0,
                d=lambda x: 1,
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            integral_dupla(
                f=lambda x, y: 1,
```

```
                a=0,
                b=1,
                pontos_e_pesos_x=n10,
                c=lambda x: 0,
                d=lambda x: 1,
                pontos_e_pesos_y=n10,
            ),
            "\n",
        )

        print("Resultado exato: 1\n")
```

Em seguida elabora-se a mesma linha de raciocínio para resolução da função para um tetraedro, definindo a função a ser integrada $f(x, y) = 1 - x - y$ e os limites em x e y sendo [0,1] e [0, $1 - x$], e, em seguida bifurca-se em três entradas possíveis para solução a depender do número de nós escolhidos.

```
# Tetraedro -------------------------------------------------------------------

        print(
            "\n\nCálculo do volume do tetraedro com vertices (0, 0, 0), (1, 0, 0), (0, 1, 0) e (0, 0, 1):\n"
        )
        print("Função a ser integrada: f(x,y) = 1-x-y\n")
        print("Intervalos de integração: \n a = 0, b = 1, c = 0, d = 1-x\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            integral_dupla(
                f=lambda x, y: 1 - x - y,
                a=0,
                b=1,
                pontos_e_pesos_x=n6,
                c=lambda x: 0,
                d=lambda x: 1 - x,
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 -->  I = ",
            integral_dupla(
                f=lambda x, y: 1 - x - y,
                a=0,
                b=1,
                pontos_e_pesos_x=n8,
                c=lambda x: 0,
                d=lambda x: 1 - x,
                pontos_e_pesos_y=n8,
            ),
        )
```

```python
        print(
            "n = 10 --> I = ",
            integral_dupla(
                f=lambda x, y: 1 - x - y,
                a=0,
                b=1,
                pontos_e_pesos_x=n10,
                c=lambda x: 0,
                d=lambda x: 1 - x,
                pontos_e_pesos_y=n10,
            ),
            "\n",
        )

        print("Resultado exato: 1/6\n")
```

3.5 Modo II

O segundo modo é dedicado a resolver o segundo exemplo descrito neste exercício como segue:

Exemplo 2 A área A da região no primeiro quadrante limitada pelos eixos e pela curva $y = 1 - x^2$ pode ser obtida por

$$A = \int_0^1 \left[\int_0^{1-x^2} dy \right] dx = \int_0^1 \left[\int_0^{\sqrt{1-y}} dx \right] dy = \frac{2}{3}$$

Calcule numericamente as duas integrais duplas acima e observe os resultados.

Figura 3: Enunciado Exemplo II

Para essa questão a área da região no primeiro quadrante limitada pelos eixos e pela curva y $= 1 - x^2$ já possibilita o modo a assumir a função a ser integrada $f(x, y) = 1$ e os intervalos [0,1]em x e $[0,1 - x]$ em y e da mesma forma que no exercício anterior, dedica-se a efetuar três contas diferentes com a única alteração do número de nós escolhidos pelo usuário.

```python
elif modo == 2:
        """Modo 2 - Resolve o Exemplo 2"""

        print(
            "\n\nA area da regiao no primeiro quadrante limitada pelos eixos e pela curva y = 1-x^2:\n"
        )
        print("Função a ser integrada: f(x,y) = 1\n")
        print("Intervalos de integração: \n a = 0, b = 1, c = 0, d = 1-x^2\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            integral_dupla(
```

```
                f=lambda x, y: 1,
                a=0,
                b=1,
                pontos_e_pesos_x=n6,
                c=lambda x: 0,
                d=lambda x: 1 - x**2,
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 -->  I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
                pontos_e_pesos_x=n8,
                c=lambda x: 0,
                d=lambda x: 1 - x**2,
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
                pontos_e_pesos_x=n10,
                c=lambda x: 0,
                d=lambda x: 1 - x**2,
                pontos_e_pesos_y=n10,
            ),
            "\n",
        )
```

Em seguida , é utilizado o mesmo método para resolução da integral dupla com ordem invertida : primeiro a integração em x e depois em y. Destaca-se que como essa alteração é não usual, embora a parte impressa em texto no início da resolução indique os valores "invertidos"entre os limites a,b,c,d ; no código que o segue a alteração é feita para que o código corra sem maiores problemas. Segue a exemplificação:

```
print(
            "\n\nA area da regiao no primeiro quadrante limitada pelos eixos e pela curva x = (1-y)^(1/2):\n"
        )
        print("Função a ser integrada: f(x,y) = 1\n")
        print("Intervalos de integração: \n a = 0, b = (1-y)^(1/2), c = 0, d = 1\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            integral_dupla(
```

```
                    f=lambda x, y: 1,
                    a= 0,
                    b= 1,
                    pontos_e_pesos_x=n6,
                    c= lambda x : 0,
                    d= lambda x: np.sqrt(1-x) , #necessário alteração da orientação
                    pontos_e_pesos_y=n6,
                ),
        )
        print(
            "n = 8 --> I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
                pontos_e_pesos_x=n8,
                c=lambda x: 0,
                d=lambda x: np.sqrt(1-x),
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
                pontos_e_pesos_x=n10,
                c=lambda x: 0,
                d=lambda x: np.sqrt(1-x),
                pontos_e_pesos_y=n10,
            ),
            "\n",
        )

        print("Resultado exato: 2/3\n")
```

3.6 Modo III

Neste exemplo, como citado abaixo, é dado uma função a qual deve-se integrar para obter o seu volume e em seguida a área da superfície.

Exemplo 3 Considere a superfície descrita por $z = e^{y/x}$, $0.1 \leq x \leq 0.5$, $x^3 \leq y \leq x^2$. Calcule a sua área e o volume da região abaixo dela (a área de uma superfície descrita por $z = f(x, y)$, $(x, y) \in R$ é igual a

$$\iint_R \sqrt{f_x(x,y)^2 + f_y(x,y)^2 + 1}\, dx\, dy \Big).$$

Figura 4: Enunciado Exemplo III

Para a integração do volume, deve-se simplesmente usar o método de integração proposto na função dada z dada nos intervalos de integração também dados, como mostrado a seguir:

```python
elif modo == 3:
        """Modo 3 - Resolve o Exemplo 3"""
        print("Area e volume da superfície descrita por:")
        print("z = e^(y/x),")
        print("0.1 <= x <= 0.5,")
        print("x^3 <= y <= x^2\n")

        print("Obs: x e y devem ser trocados para ordem correta de integração\n")

        # Volume ----------------------------------------------------------------

        print("Cálculo do volume da superfície descrita:\n")
        print("Função a ser integrada: f(x,y) = e^(y/x)\n")
        print("Intervalos de integração: \n a = 0.1, b = 0.5, c = x^3, d = x^2\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            integral_dupla(
                f=lambda x, y: np.e ** (y / x),
                a=0.1,
                b=0.5,
                pontos_e_pesos_x=n6,
                c=lambda x: x**3,
                d=lambda x: x**2,
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 -->  I = ",
            integral_dupla(
                f=lambda x, y: np.e ** (y / x),
                a=0.1,
                b=0.5,
                pontos_e_pesos_x=n8,
                c=lambda x: x**3,
                d=lambda x: x**2,
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            integral_dupla(
                f=lambda x, y: np.e ** (y / x),
                a=0.1,
                b=0.5,
```

```
                    pontos_e_pesos_x=n10,
                    c=lambda x: x**3,
                    d=lambda x: x**2,
                    pontos_e_pesos_y=n10,
                ),
                "\n",
            )
```

Para integração na superfície devemos primeiro chegar na fórmula dada ao final do enunciado, o qual é compostos pelas derivadas parciais da função original e após isso integrar a expressão resultante nos intervalos dados na seguinte forma:

```
# Superficie ------------------------------------------------------------------
        print("Cálculo da superfície descrita:\n")
        print(
            "Função a ser integrada: f(x,y) = ( (e^(x/y)/y)^2 + (-x*e^(x/y)/y^2)^2 + 1 )^(1/2)\n"
        )
        print("Intervalos de integração: \n a = 0.1, b = 0.5, c = x^3, d = x^2\n")
        print("Resultados: ")
        print(
            "n = 6 --> I = ",
            integral_dupla(
                f=lambda x, y: (
                    (np.e ** (y / x) / x) ** 2
                    + (-y * np.e ** (y / x) / x**2) ** 2
                    + 1
                )
                ** (1 / 2),
                a=0.1,
                b=0.5,
                pontos_e_pesos_x=n6,
                c=lambda x: x**3,
                d=lambda x: x**2,
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 --> I = ",
            integral_dupla(
                f=lambda x, y: (
                    (np.e ** (y / x) / x) ** 2
                    + (-y * np.e ** (y / x) / x**2) ** 2
                    + 1
                )
                ** (1 / 2),
                a=0.1,
                b=0.5,
                pontos_e_pesos_x=n8,
```

```
                c=lambda x: x**3,
                d=lambda x: x**2,
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            integral_dupla(
                f=lambda x, y: (
                    (np.e ** (y / x) / x) ** 2
                    + (-y * np.e ** (y / x) / x**2) ** 2
                    + 1
                )
                ** (1 / 2),
                a=0.1,
                b=0.5,
                pontos_e_pesos_x=n10,
                c=lambda x: x**3,
                d=lambda x: x**2,
                pontos_e_pesos_y=n10,
            ),
            "\n",
        )
```

3.7 Modo IV

Exemplo 4 Considere uma região fechada R do plano xy e seja γ uma reta no mesmo plano que não intercepta o interior de R. O volume V do sólido de revolução obtido da rotação da região R em torno de γ é igual a

$$V = 2\pi \iint_R d_\gamma(x, y)\, dx\, dy$$

onde $d_\gamma(x, y)$ é a distância do ponto (x, y) à reta γ. Use esta expressão para calcular o volume da calota esférica de altura $\frac{1}{4}$ da esfera de raio 1, e o volume do sólido de revolução obtido da rotação da região, em torno do eixo y, delimitada por $x = 0$, $x = e^{-y^2}$, $y = -1$ e $y = 1$.

Figura 5: Enunciado Exemplo III

Neste último modo, dedica-se a encontrar as soluções em duas etapas: a primeira encontra-se o volume da calota esférica pedida, com 1/4 de altura do raio e a segunda em calcular o volume do sólido de revolução na rotação no espaço descrito.

Em primeiro, calculando o volume da calota, integra-se a função f(x,y)= y nos intervalos [3/4, 1] - por conta da altura de 1/4 do raio - e em y de zero à equação da circunferência para a área inicial da esfera utilizada. Então utiliza-se o método de integração:

```
elif modo == 4:
        """Modo 4 - Resolve o Exemplo 4"""

        # Volume da calota esferica
    ____________________________________________________________
```

```python
        print(
            "\n\nCalculo do volume da calota esferica de altura 1/4 da esfera de raio 1\n"
        )
        print("Função a ser integrada: f(x,y) = y\n")
        print("Intervalos de integração: \n a = 3/4, b = 1, c = 0, d = (1-x^2)^(1/2)\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            2
            * np.pi
            * integral_dupla(
                f=lambda x, y: y,
                a=3 / 4,
                b=1,
                pontos_e_pesos_x=n6,
                c=lambda x: 0,
                d=lambda x: (1 - x**2) ** (1 / 2),
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 -->  I = ",
            2
            * np.pi
            * integral_dupla(
                f=lambda x, y: y,
                a=3 / 4,
                b=1,
                pontos_e_pesos_x=n8,
                c=lambda x: 0,
                d=lambda x: (1 - x**2) ** (1 / 2),
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            2
            * np.pi
            * integral_dupla(
                f=lambda x, y: y,
                a=3 / 4,
                b=1,
                pontos_e_pesos_x=n10,
                c=lambda x: 0,
                d=lambda x: (1 - x**2) ** (1 / 2),
                pontos_e_pesos_y=n10,
            ),
            "\n",
        )
```

Para a segunda parte integramos a mesma função f(x,y) porém a mudança será nos limites de integração escolhidos, uma vez que estes serão delimitados pelas funções que parametrizam a rotação do sólido formado. Os limites ficam respectivamente [-1,1] em x e [0, e^{-y^2}] em y.

```python
# Volume do solido de revolucao ---------------------------------------------------------
        print(
            "\n\nCalculo do volume do solido de revolucao obtido da rotacao da regiao x=0, x=e^(-y^2), y=-1, y=1 em torno do eixo y\n"
        )
        print("Função a ser integrada: f(x,y) = y\n")
        print("Intervalos de integração: \n a = -1, b = 1, c = 0, d = e^(-y^2)\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            2
            * np.pi
            * integral_dupla(
                f=lambda x, y: y,
                a=-1,
                b=1,
                pontos_e_pesos_x=n6,
                c=lambda x: 0,
                d=lambda x: np.exp(-(x**2)),
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 -->  I = ",
            2
            * np.pi
            * integral_dupla(
                f=lambda x, y: y,
                a=-1,
                b=1,
                pontos_e_pesos_x=n8,
                c=lambda x: 0,
                d=lambda x: np.exp(-(x**2)),
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            2
            * np.pi
            * integral_dupla(
```

```
                    f=lambda x, y: y,
                    a=-1,
                    b=1,
                    pontos_e_pesos_x=n10,
                    c=lambda x: 0,
                    d=lambda x: np.exp(-(x**2)),
                    pontos_e_pesos_y=n10,
                ),
                "\n",
            )
```

4 Conclusão

4.1 Resultados

Seguem abaixo os resultados e comentários sobre os devidos outputs testados para verificação dos códigos apresentados neste exercício programa.

4.1.1 Modo I

Para a verificação deste modo, comparou-se o resultado obtido nas três operações com diferentes número de nós com aquele fornecido no enunciado, o qual está escrito ao final do modo também. No modo um, tanto 6 quanto 8 quanto 10 nós coincidiram com o resultado exato escrito, a única diferença está no grau de precisão, ou seja o número de significativos de cada resposta mas todas chegam na resposta exata do exercício.

```
Modo: 1


Cálculo do volume do cubo cujas arestas tem comprimento 1:

Função a ser integrada: f(x,y) = 1

Intervalos de integração:
 a = 0, b = 1, c = 0, d = 1

Resultados:
n = 6 -->  I =  1.0
n = 8 -->  I =  1.0
n = 10 --> I =  0.9999999999999998

Resultado exato: 1



Cálculo do volume do tetraedro com vertices (0, 0, 0), (1, 0, 0), (0, 1, 0) e (0, 0, 1):

Função a ser integrada: f(x,y) = 1-x-y

Intervalos de integração:
 a = 0, b = 1, c = 0, d = 1-x

Resultados:
n = 6 -->  I =  0.16666666666666666
n = 8 -->  I =  0.16666666666666669
n = 10 --> I =  0.16666666666666666

Resultado exato: 1/6
```

Figura 6: Output total do modo I

4.1.2 Modo II

Para a verificação deste modo, utilizou-se o mesmo método que para o exercício anterior comparou-se o resultado obtido nas três operações com diferentes número de nós com aquele fornecido no enunciado, o qual está indicado na saída ao final do modo também. No modo um, tanto 6 quanto 8 quanto 10 nós coincidiram com o resultado exato mencionado, a única diferença está no grau de precisão, ou seja o número de significativos de cada resposta mas todas chegam na resposta exata do exercício.

Além da confiabilidade do código, é necessário comentar que durante a segunda resolução de duplas integrais é possível observar que a precisão do resultado é ligeiramente inferior à primeira. Isso se deve majoritariamente à utilização da expressão para o limite de x conter uma raiz quadrada ($x = \sqrt{1-y}$), o que acaba propagando uma imprecisão maior que a expressão para a sua integral equivalente que não utiliza ($y = 1 - x^2$)

```
Modo: 2


A area da regiao no primeiro quadrante limitada pelos eixos e pela curva y = 1-x^2:

Função a ser integrada: f(x,y) = 1

Intervalos de integração:
 a = 0, b = 1, c = 0, d = 1-x^2

Resultados:
n = 6 -->  I =  0.6666666666666666
n = 8 -->  I =  0.6666666666666667
n = 10 --> I =  0.6666666666666666



A area da regiao no primeiro quadrante limitada pelos eixos e pela curva x = (1-y)^(1/2):

Função a ser integrada: f(x,y) = 1

Intervalos de integração:
 a = 0, b = (1-y)^(1/2), c = 0, d = 1

Resultados:
n = 6 -->  I =  0.6670464379156136
n = 8 -->  I =  0.6668355801001764
n = 10 --> I =  0.6667560429365088

Resultado exato: 2/3
```

Figura 7: Output total do modo II

4.1.3 Modo III

Para a conclusão final da confiança no código deste modo, como ao contrário dos anteriores não havia resposta exata disponível, lançou-se mão de calculadoras online utilizadas tipicamente para problemas em cálculo - no caso o Wolfram Alpha.

A seguir observa-se os devidos resultados tanto do código tanto da calculadora. Os outputs são coerentes e podemos considerar o código confiável. Vale a ressalvá de que na primeira parte aparece uma pequena parte imaginária na resolução pela calculadora, porém sendo está de ordem de grandeza inferior à 10^{-16} podemos considerar ela desprezível.

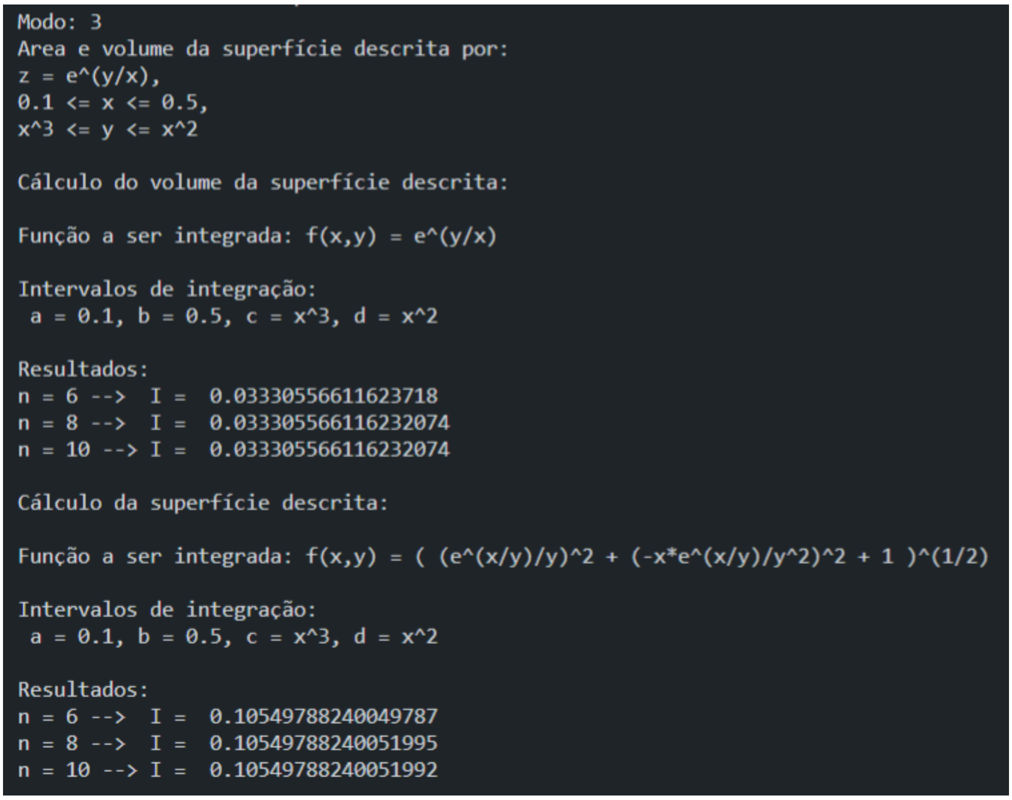

Figura 8: Output dos valores obtidos no modo III no código

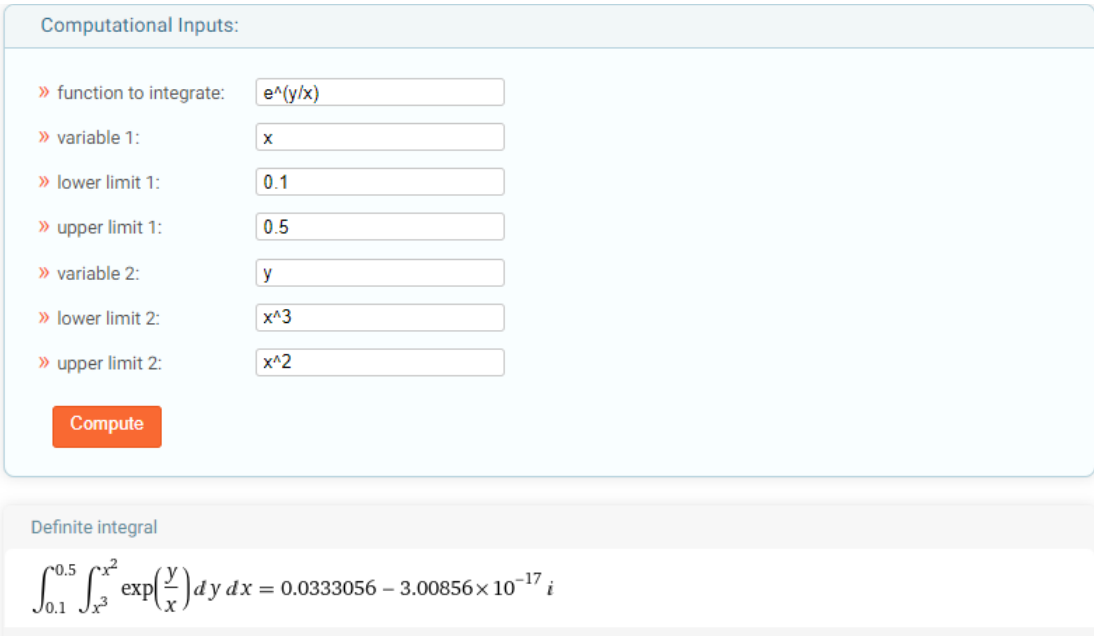

Figura 9: Output de soluções no Wolfram Alpha para primeira parte da solução

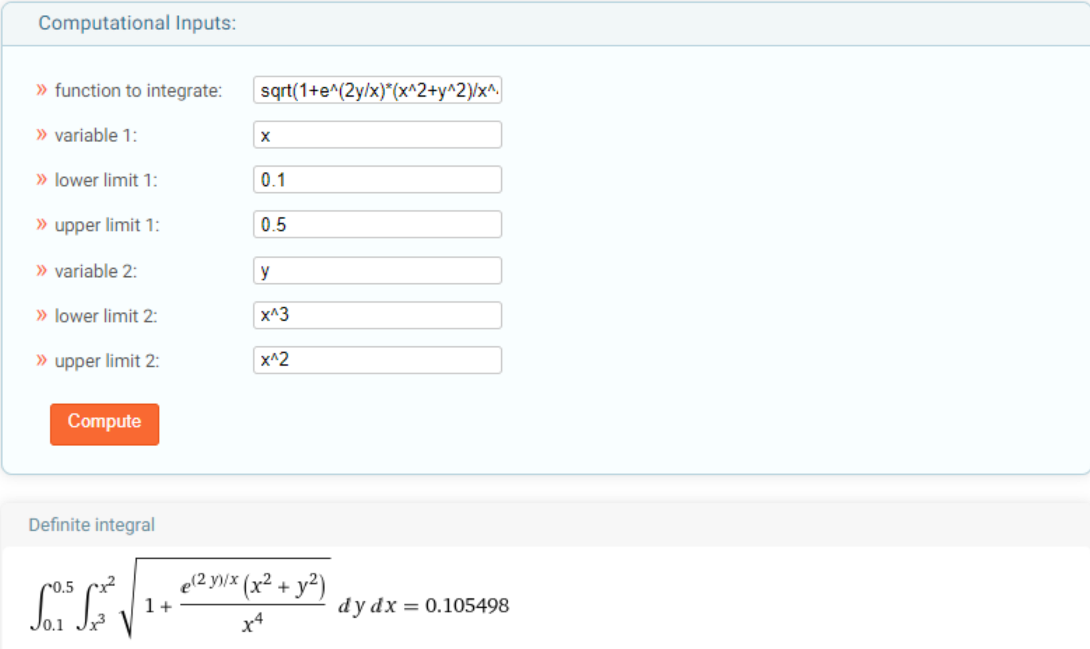

Figura 10: Output de soluções no Wolfram Alpha para segunda parte da solução

4.1.4 Modo IV

Para este último teste, foram utilizados os mesmos métodos que anteriormente, comparando os resultados obtidos pelo código com aqueles efetuados em ferramentas online para resolução de integrações duplas em regiões do espaço. Mais uma vez observa-se que os resultados estão coerentes e pode-se atestar a confiabilidade do código.

Uma conclusão além da confiabilidade do método é que por se tratar de uma função polinomial e um limite também polinomial o resultado dá exato como o esperado, como mencionado no próprio enunciado.

```
Modo: 4


Calculo do volume da calota esferica de altura 1/4 da esfera de raio 1

Função a ser integrada: f(x,y) = y

Intervalos de integração:
 a = 3/4, b = 1, c = 0, d = (1-x^2)^(1/2)

Resultados:
n = 6 --> I =  0.1799870791119152
n = 8 --> I =  0.17998707911191522
n = 10 --> I =  0.17998707911191525



Calculo do volume do solido de revolucao obtido da rotacao da regiao x=0, x=e^(-y^2), y=-1, y=1 em torno do eixo y

Função a ser integrada: f(x,y) = y

Intervalos de integração:
 a = -1, b = 1, c = 0, d = e^(-y^2)

Resultados:
n = 6 --> I =  3.7581650328967093
n = 8 --> I =  3.7582492624394384
n = 10 --> I =  3.7582496332093873
```

Figura 11: Output de todos os resultados para o modo IV no código em Pyhton

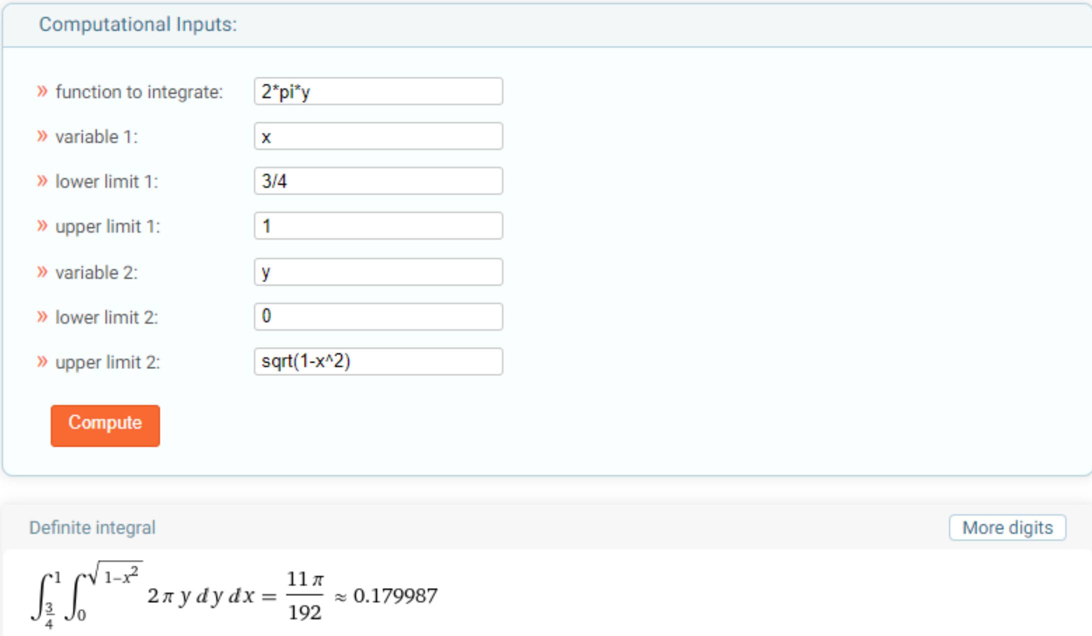

Figura 12: Resultado obtido na resolução da primeira parte do modo no Wolfram

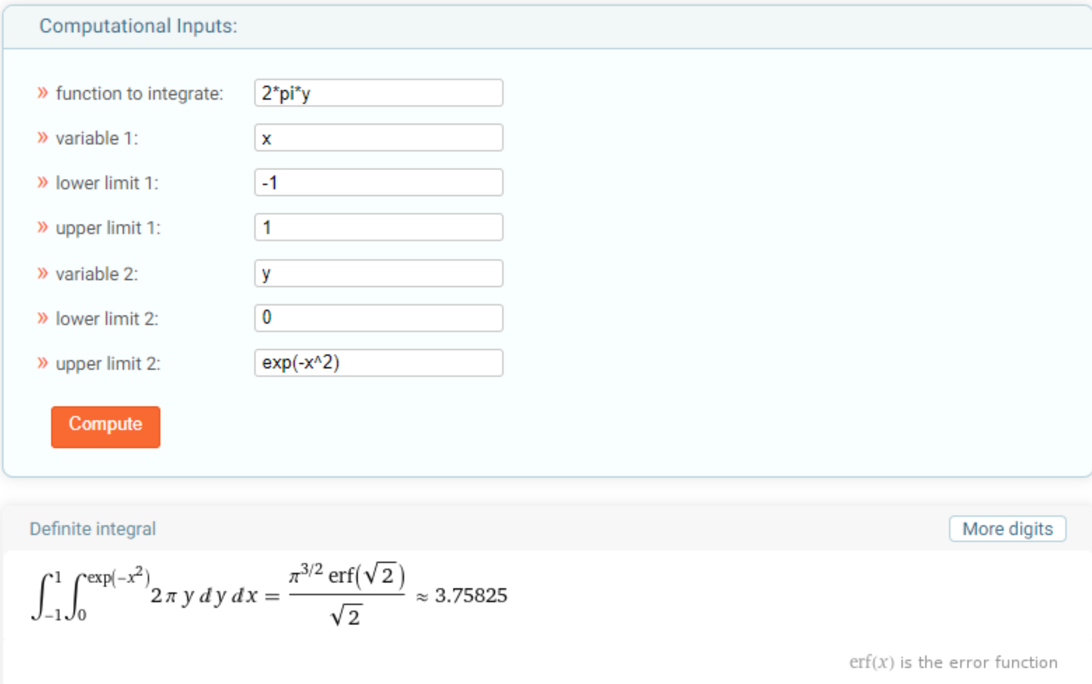

Figura 13: Resultado obtido na resolução da segunda parte do modo no Wolfram

5 Referências

VALLE,Marcos Eduardo. **Aula 24 Quadratura Gaussiana.**.Departamento de Matemática Aplicada Instituto de Matemática, Estatística e Computação Científica Universidade Estadual de Campinas Disponível em : <https://www.ime.unicamp.br/ valle/Teaching/2015/MS211/Aula24.pdf> Acesso em 31 de maio,2022.

OLIVEIRA,Maria Luísa Bambozzi de. **Integração Numérico**.27 de Outubro, 2010 e 8 de Novembro, 2010. Disponível em
<https://sites.icmc.usp.br/marialuisa/cursos201002/integracao$_n$*umerica.pdf* > .
Acesso em 30 de maio, 2022.

Hjorth-Jensen,Morten . **Computational Physics Lectures: Numerical integration, from Newton-Cotes quadrature to Gaussian quadrature** . Department of Physics, University of Oslo. 23 de agosto, 2017. Disponível em:
<http://compphysics.github.io/ComputationalPhysics/doc/pub/integrate/html/integrate.html>.
Acesso em: 31 maio 2022.

Gauss quadrature formula.Encyclopedia of Mathematics. Disponível em:
<http://encyclopediaofmath.org/index.php?title=Gauss$_q$*uadrature$_f$ormulaoldid* = 43647 > .
Acesso em: 30 maio 2022.

INTEGRAÇÃO NUMÉRICA - QUADRATURA DE GAUSS LEGENDRE.Métodos Numéricos. Youtube. 21 de novembro de 2020. 22:33. Disponível em: <https://www.youtube.com/watch?v=6W5Y2JWQnUg>. Acesso em: 31 de Maio de 2022.

Calculadora de Integrais Duplas - Wolfram Alpha. Disponível em:
<https://www.wolframalpha.com/input?i=double+integral>. Acesso em: 2 junho 2022.

NumPy documentation — NumPy v1.22 Manual. Numpy.org. Disponível em: <https://numpy.org/doc/stable/index.html>. Acesso em: 2 maio 2022.

Quadratura Gaussiana + Algoritmo em Python.Produção: Sidnei fc. Youtube. 2021. 27:49. Disponível em : <https://www.youtube.com/watch?v=bu8trr9Qm1Y> Acesso em 01/06/2022

A Apêndices

A.1 Main.py

```python
"""
EP2 - Formulas de Integracao Numerica de Gauss

Nome: Laura do Prado Gonçalves Pinto
NUSP: 11819960

Nome: Mateus Stano Junqueira
NUSP: 11804845
"""

import numpy as np

# pontos e pesos pré estabelecidos
# na forma de n = [(x_j,w_j),...]
n6 = [
    (0.2386191860831969086305017, 0.4679139345726910473898703),
    (0.6612093864662645136613996, 0.3607615730481386075698335),
    (0.9324695142031520278123016, 0.1713244923791703450402961),
    (-0.2386191860831969086305017, 0.4679139345726910473898703),
    (-0.6612093864662645136613996, 0.3607615730481386075698335),
    (-0.9324695142031520278123016, 0.1713244923791703450402961),
]
n8 = [
    (0.1834346424956498049394761, 0.3626837833783619829651504),
    (0.5255324099163289858177390, 0.3137066458778872873379622),
    (0.7966664774136267395915539, 0.2223810344533744705443560),
    (0.9602898564975362316835609, 0.1012285362903762591525314),
    (-0.1834346424956498049394761, 0.3626837833783619829651504),
    (-0.5255324099163289858177390, 0.3137066458778872873379622),
    (-0.7966664774136267395915539, 0.2223810344533744705443560),
    (-0.9602898564975362316835609, 0.1012285362903762591525314),
]
n10 = [
    (0.1488743389816312108848260, 0.2955242247147528701738930),
    (0.4333953941292471907992659, 0.2692667193099963550912269),
    (0.6794095682990244062343274, 0.2190863625159820439955349),
    (0.8650633666889845107320967, 0.1494513491505805931457763),
    (0.9739065285171717200779640, 0.0666713443086881375935688),
    (-0.1488743389816312108848260, 0.2955242247147528701738930),
    (-0.4333953941292471907992659, 0.2692667193099963550912269),
    (-0.6794095682990244062343274, 0.2190863625159820439955349),
    (-0.8650633666889845107320967, 0.1494513491505805931457763),
    (-0.9739065285171717200779640, 0.0666713443086881375935688),
]

# Calculo da integral dupla
```

```python
def integral_dupla(f, a, b, pontos_e_pesos_x, c, d, pontos_e_pesos_y):
    """
    f: função de x e y a ser integrada
    a: limite inferior do intervalo de integração em x
    b: limite superior do intervalo de integração em x
    pontos_e_pesos_x: pontos e pesos para o método de Gauss em x (deve ser da forma [(ponto1, peso1), (ponto2, peso2), ...].)
    c: limite inferior do intervalo de integração em y (função de x)
    d: limite superior do intervalo de integração em y (função de x)
    pontos_e_pesos_y: pontos e pesos para o método de Gauss em y (deve ser da forma [(ponto1, peso1), (ponto2, peso2), ...].)

    """

    I = 0
    for x_i, w_i in pontos_e_pesos_x:
        # troca de variavel de x para o intervalo [a,b]
        x_i = (a + b) / 2 + (b - a) * x_i / 2
        F = 0
        for y_ij, w_ij in pontos_e_pesos_y:
            # troca de variavel de y para o intervalo [c(x_i),d(x_i)]
            y_ij = (c(x_i) + d(x_i)) / 2 + (d(x_i) - c(x_i)) * y_ij / 2
            F += w_ij * f(x_i, y_ij) * (d(x_i) - c(x_i)) / 2
        I += w_i * F * (b - a) / 2

    return I


def main():
    print("Escolha um modo:")
    print("Modo 0 - Calculo de integral dupla para uma função e intervalos quaisquer")
    print("Modo 1 - Resolve o Exemplo 1")
    print("Modo 2 - Resolve o Exemplo 2")
    print("Modo 3 - Resolve o Exemplo 3")
    print("Modo 4 - Resolve o Exemplo 4")
    modo = int(input("Modo: "))

    if modo == 0:
        """Modo 0 - Calculo de integral dupla para uma função e intervalos quaisquer"""

        print("Calculo de integral dupla da forma dydx")
        f = input("Função f(x,y) a ser integrada: ")
        f_str = ""
        for char in f:  # Converte a função para uma string utilizavel em eval()
            if char == "^":
```

```
            f_str += "**"
        else:
            f_str += char
    f_final = lambda x, y: eval(
        f
    )  # Cria uma funcao lambda que recebe x e y e retorna o valor da função f(x,y)

    a = float(
        input(
            "Limite inferior do intervalo de integração da integral exterior (dx): "
        )
    )

    b = float(
        input(
            "Limite superior do intervalo de integração da integral exterior (dx): "
        )
    )

    nx = int(input("Número de subintervalos (nós) para x (opções 6, 8 ou 10): "))
    if nx == 6:
        nx = n6
    elif nx == 8:
        nx = n8
    elif nx == 10:
        nx = n10
    else:
        raise ValueError("Número de subintervalos inválido")

    c = input(
        "Limite inferior do intervalo de integração da integral exterior (dy): "
    )
    c_str = ""
    for char in c:  # Converte a função para uma string utilizavel em eval()
        if char == "^":
            c_str += "**"
        else:
            c_str += char
    c_final = lambda x: eval(
        c_str
    )  # Cria uma funcao lambda que recebe x e retorna o valor da função c(x)

    d = input(
```

```
            "Limite superior do intervalo de integração da integral exterior (dy): "
        )
        d_str = ""
        for char in d:  # Converte a função para uma string utilizavel em eval()
            if char == "^":
                d_str += "**"
            else:
                d_str += char
        d_final = lambda x: eval(
            d_str
        )  # Cria uma funcao lambda que recebe x e retorna o valor da função d(x)

        ny = int(input("Número de subintervalos (nós) para y (opções 6, 8 ou 10): "))
        if ny == 6:
            ny = n6
        elif ny == 8:
            ny = n8
        elif ny == 10:
            ny = n10
        else:
            raise ValueError("Número de subintervalos inválido")

        print("\nIntegral da forma dydx:")
        print(
            "I = ",
            integral_dupla(f_final, a, b, nx, c_final, d_final, ny),
            "\n",
        )

    elif modo == 1:
        """Modo 1 - Resolve o Exemplo 1"""

        # Cubo --------------------------------------------------------------------------

        print("\n\nCálculo do volume do cubo cujas arestas tem comprimento 1:\n")
        print("Função a ser integrada: f(x,y) = 1\n")
        print("Intervalos de integração: \n a = 0, b = 1, c = 0, d = 1\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
```

```
                pontos_e_pesos_x=n6,
                c=lambda x: 0,
                d=lambda x: 1,
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 -->  I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
                pontos_e_pesos_x=n8,
                c=lambda x: 0,
                d=lambda x: 1,
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
                pontos_e_pesos_x=n10,
                c=lambda x: 0,
                d=lambda x: 1,
                pontos_e_pesos_y=n10,
            ),
            "\n",
        )

        print("Resultado exato: 1\n")

        # Tetraedro -----------------------------------------------------------------

        print(
            "\n\nCálculo do volume do tetraedro com vertices (0, 0, 0), (1, 0, 0), (0, 1, 0) e (0, 0, 1):\n"
        )
        print("Função a ser integrada: f(x,y) = 1-x-y\n")
        print("Intervalos de integração: \n a = 0, b = 1, c = 0, d = 1-x\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            integral_dupla(
                f=lambda x, y: 1 - x - y,
                a=0,
                b=1,
```

```
                pontos_e_pesos_x=n6,
                c=lambda x: 0,
                d=lambda x: 1 - x,
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 --> I = ",
            integral_dupla(
                f=lambda x, y: 1 - x - y,
                a=0,
                b=1,
                pontos_e_pesos_x=n8,
                c=lambda x: 0,
                d=lambda x: 1 - x,
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            integral_dupla(
                f=lambda x, y: 1 - x - y,
                a=0,
                b=1,
                pontos_e_pesos_x=n10,
                c=lambda x: 0,
                d=lambda x: 1 - x,
                pontos_e_pesos_y=n10,
            ),
            "\n",
        )

        print("Resultado exato: 1/6\n")

    elif modo == 2:
        """Modo 2 - Resolve o Exemplo 2"""

        print(
            "\n\nA area da regiao no primeiro quadrante limitada pelos eixos e pela curva y = 1-x^2:\n"
        )
        print("Função a ser integrada: f(x,y) = 1\n")
        print("Intervalos de integração: \n a = 0, b = 1, c = 0, d = 1-x^2\n")
        print("Resultados: ")
        print(
            "n = 6 --> I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
```

```
                pontos_e_pesos_x=n6,
                c=lambda x: 0,
                d=lambda x: 1 - x**2,
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 -->  I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
                pontos_e_pesos_x=n8,
                c=lambda x: 0,
                d=lambda x: 1 - x**2,
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
                pontos_e_pesos_x=n10,
                c=lambda x: 0,
                d=lambda x: 1 - x**2,
                pontos_e_pesos_y=n10,
            ),
            "\n",
        )

        print(
            "\n\nA area da regiao no primeiro quadrante limitada pelos eixos e pela curva x = (1-y)^(1/2):\n"
        )
        print("Função a ser integrada: f(x,y) = 1\n")
        print("Intervalos de integração: \n a = 0, b = (1-y)^(1/2), c = 0, d = 1\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a= 0,
                b= 1,
                pontos_e_pesos_x=n6,
                c= lambda x : 0,
                d= lambda x: np.sqrt(1-x) , #necessário alteração da orientação
                pontos_e_pesos_y=n6,
```

```
            ),
        )
        print(
            "n = 8 -->  I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
                pontos_e_pesos_x=n8,
                c=lambda x: 0,
                d=lambda x: np.sqrt(1-x),
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            integral_dupla(
                f=lambda x, y: 1,
                a=0,
                b=1,
                pontos_e_pesos_x=n10,
                c=lambda x: 0,
                d=lambda x: np.sqrt(1-x),
                pontos_e_pesos_y=n10,
            ),
            "\n",
        )

        print("Resultado exato: 2/3\n")

    elif modo == 3:
        """Modo 3 - Resolve o Exemplo 3"""
        print("Area e volume da superfície descrita por:")
        print("z = e^(y/x),")
        print("0.1 <= x <= 0.5,")
        print("x^3 <= y <= x^2\n")

        print("Obs: x e y devem ser trocados para ordem correta de integração\n")

        # Volume ------------------------------------------------------------------------

        print("Cálculo do volume da superfície descrita:\n")
        print("Função a ser integrada: f(x,y) = e^(y/x)\n")
        print("Intervalos de integração: \n a = 0.1, b = 0.5, c = x^3, d = x^2\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            integral_dupla(
```

```
                f=lambda x, y: np.e ** (y / x),
                a=0.1,
                b=0.5,
                pontos_e_pesos_x=n6,
                c=lambda x: x**3,
                d=lambda x: x**2,
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 -->  I = ",
            integral_dupla(
                f=lambda x, y: np.e ** (y / x),
                a=0.1,
                b=0.5,
                pontos_e_pesos_x=n8,
                c=lambda x: x**3,
                d=lambda x: x**2,
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            integral_dupla(
                f=lambda x, y: np.e ** (y / x),
                a=0.1,
                b=0.5,
                pontos_e_pesos_x=n10,
                c=lambda x: x**3,
                d=lambda x: x**2,
                pontos_e_pesos_y=n10,
            ),
            "\n",
        )

        # Superficie -----------------------------------------------------------------------
        print("Cálculo da superfície descrita:\n")
        print(
            "Função a ser integrada: f(x,y) = ( (e^(x/y)/y)^2 + (-x*e^(x/y)/y^2)^2 + 1 )^(1/2)\n"
        )
        print("Intervalos de integração: \n a = 0.1, b = 0.5, c = x^3, d = x^2\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            integral_dupla(
                f=lambda x, y: (
                    (np.e ** (y / x) / x) ** 2
```

```
                    + (-y * np.e ** (y / x) / x**2) ** 2
                    + 1
                )
                ** (1 / 2),
                a=0.1,
                b=0.5,
                pontos_e_pesos_x=n6,
                c=lambda x: x**3,
                d=lambda x: x**2,
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 -->  I = ",
            integral_dupla(
                f=lambda x, y: (
                    (np.e ** (y / x) / x) ** 2
                    + (-y * np.e ** (y / x) / x**2) ** 2
                    + 1
                )
                ** (1 / 2),
                a=0.1,
                b=0.5,
                pontos_e_pesos_x=n8,
                c=lambda x: x**3,
                d=lambda x: x**2,
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            integral_dupla(
                f=lambda x, y: (
                    (np.e ** (y / x) / x) ** 2
                    + (-y * np.e ** (y / x) / x**2) ** 2
                    + 1
                )
                ** (1 / 2),
                a=0.1,
                b=0.5,
                pontos_e_pesos_x=n10,
                c=lambda x: x**3,
                d=lambda x: x**2,
                pontos_e_pesos_y=n10,
            ),
            "\n",
        )

    elif modo == 4:
        """Modo 4 - Resolve o Exemplo 4"""

```

```python
        # Volume da calota esferica ----------------------------------------------------
        print(
            "\n\nCalculo do volume da calota esferica de altura 1/4 da esfera de raio 1\n"
        )
        print("Função a ser integrada: f(x,y) = y\n")
        print("Intervalos de integração: \n a = 3/4, b = 1, c = 0, d = (1-x^2)^(1/2)\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            2
            * np.pi
            * integral_dupla(
                f=lambda x, y: y,
                a=3 / 4,
                b=1,
                pontos_e_pesos_x=n6,
                c=lambda x: 0,
                d=lambda x: (1 - x**2) ** (1 / 2),
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 -->  I = ",
            2
            * np.pi
            * integral_dupla(
                f=lambda x, y: y,
                a=3 / 4,
                b=1,
                pontos_e_pesos_x=n8,
                c=lambda x: 0,
                d=lambda x: (1 - x**2) ** (1 / 2),
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            2
            * np.pi
            * integral_dupla(
                f=lambda x, y: y,
                a=3 / 4,
                b=1,
                pontos_e_pesos_x=n10,
                c=lambda x: 0,
                d=lambda x: (1 - x**2) ** (1 / 2),
                pontos_e_pesos_y=n10,
            ),
```

```
            "\n",
        )

        # Volume do solido de revolucao ------------------------------------------------------------
        print(
            "\n\nCalculo do volume do solido de revolucao obtido da rotacao da regiao x=0, x=e^(-y^2), y=-1, y=1 em torno do eixo y\n"
        )
        print("Função a ser integrada: f(x,y) = y\n")
        print("Intervalos de integração: \n a = -1, b = 1, c = 0, d = e^(-y^2)\n")
        print("Resultados: ")
        print(
            "n = 6 -->  I = ",
            2
            * np.pi
            * integral_dupla(
                f=lambda x, y: y,
                a=-1,
                b=1,
                pontos_e_pesos_x=n6,
                c=lambda x: 0,
                d=lambda x: np.exp(-(x**2)),
                pontos_e_pesos_y=n6,
            ),
        )
        print(
            "n = 8 -->  I = ",
            2
            * np.pi
            * integral_dupla(
                f=lambda x, y: y,
                a=-1,
                b=1,
                pontos_e_pesos_x=n8,
                c=lambda x: 0,
                d=lambda x: np.exp(-(x**2)),
                pontos_e_pesos_y=n8,
            ),
        )
        print(
            "n = 10 --> I = ",
            2
            * np.pi
            * integral_dupla(
                f=lambda x, y: y,
                a=-1,
                b=1,
                pontos_e_pesos_x=n10,
```

```python
                        c=lambda x: 0,
                        d=lambda x: np.exp(-(x**2)),
                        pontos_e_pesos_y=n10,
                    ),
                    "\n",
            )


if __name__ == "__main__":
    main()
```